DAM
Desarrollo de Aplicaciones Multiplataforma
2º Curso

AD
Acceso a Datos

UD 2
Manejo de Ficheros
Parte 5 - CSV

IES BALMIS
Dpto Informática
Curso 2019-2020
Versión 1 (03/2019)

UD2 – Manejo de ficheros

ÍNDICE

13. Ficheros CSV

Los archivos **CSV** (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: Chile, Perú, Argentina, España, Brasil...) y las filas por saltos de línea.

El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. Estos puntos deben indicarse muchas veces al abrir el archivo, por ejemplo, con una hoja de cálculo.

El formato **CSV** no está estandarizado. La idea básica de separar los campos con una coma es muy clara, pero se vuelve complicada cuando el valor del campo también contienen comillas dobles o saltos de línea. Las implementaciones de CSV pueden no manejar esos datos, o usar comillas de otra clase para envolver el campo. Pero esto no resuelve el problema: algunos campos también necesitan embeber estas comillas, así que las implementaciones de CSV pueden incluir caracteres o secuencias de escape.

Además, se suele crear un primer registro de cabeceras que indica el nombre de los campos de cada columna:

```
idpro,descrip,precio
1,productoA,50.0
2,productoB,90.0
3,productoC,40.0
```

Separadores

Además, el término "CSV" también denota otros formatos de valores separados por delimitadores que usan delimitadores diferentes a la coma (como los valores separados por tabuladores). Un delimitador que no está presente en los valores de los campos (como un tabulador) mantiene el formato simple. Estos archivos separados por delimitadores alternativos reciben en algunas ocasiones la extensión aunque este uso sea incorrecto. Esto puede causar problemas en el intercambio de datos, por ello muchas aplicaciones que usan archivos CSV tienen opciones para cambiar el carácter delimitador.

Por ejemplo, un separador muy utilizado es el carácter `' | '`

```
idpro|descrip|precio
1|productoA|50.0
2|productoB|90.0
3|productoC|40.0
```

Normalmente, guardamos en CSV datos que podemos representar en tablas:

Año	Marca	Modelo	Descripción	Precio
1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture	Extended Edition	4900.00
1999	Chevy	Venture	Extended Edition, Very Large	5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

La tabla de arriba puede ser representada de la siguiente forma en CSV:

```
Año,Marca,Modelo,Descripción,Precio
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,Venture,Extended Edition,4900.00
1999,Chevy,Venture,"Extended Edition, Very Large",5000.00
1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof, loaded",4799.00
```

Este tipo de archivos puede abrirse y crearse desde hojas de cálculo. Crea un archivo denominado **coches.csv** con el siguiente contenido y ábrelo con la hoja de cálculo Calc.

```
Año,Marca,Modelo,Descripción,Precio
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,Venture,Extended Edition,4900.00
1999,Chevy,Venture,"Extended Edition, Very Large",5000.00
1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof, loaded",4799.00
```

13.1.- CSV desde Java

Para crear archivos CSV desde Java de forma sencilla disponemos de una librería denominada "**javacsv.jar**".

Java CSV componente
https://www.csvreader.com/java_csv.php

Dispondremos de dos clases, una para leer "**CsvReader**" y otra para escribir "**CsvWriter**".

Java CSV componente - Doc
http://javacsv.sourceforge.net/

13.2.- Crear CSV desde Java

Vamos a crear el siguiente archivo CSV denominado "**users.csv**":

```
id|name
1|Juan
2|Pedro
3|Inés
```

Tendremos una parte para crear el objecto de la clase **CsvWriter** y luego otra para escribir la cabecera y los registros.

```
public static void main(String[] args) {

  String outputFile = "./datos/users.csv";

  // Antes de abrir el fichero comprobamos si existe
  boolean alreadyExists = new File(outputFile).exists();

  try {

    CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '|');

    csvOutput.setDelimiter('|');      // No sería necesario porque ya se le ha indicado
    csvOutput.setRecordDelimiter('\n');  // Es el valor por defecto

    // Si ya existe el fichero no se necesita escrbir las cabeceras
    if (!alreadyExists)    {
      csvOutput.write("id");
      csvOutput.write("name");
      csvOutput.endRecord();
    }
    // ELSE asume que como ya existe tiene las cabeceras


    // Escribe unos registros
    csvOutput.write("1");
    csvOutput.write("Juan");
    csvOutput.endRecord();

    csvOutput.write("2");
    csvOutput.write("Pedro");
    csvOutput.endRecord();

    csvOutput.write("3");
    csvOutput.write("Inés");
    csvOutput.endRecord();

    csvOutput.close();

  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

Por defecto se usará la codificación "**ISO-8859-1**" Si deseamos utilizar la codificación "**UTF-8**" deberemos indicarlo. En esto caso, utilizaremos otro constructor:

```
...
  // ****************************
  // Para añadir indicando Charset
  // ****************************
  OutputStream fw = new FileOutputStream("./datos/users.csv", true);
  CsvWriter csvOutput = new CsvWriter(fw, '|', Charset.forName("UTF-8"));
...
```

Si solo lo vamos a crear (es decir, no añadimos porque se sobreescribe si existe) es un poco más simple:

```
...
  // ****************************
  // Para crear indicando Charset
  // ****************************
  CsvWriter csvOutput = new CsvWriter("./datos/users.csv",'|',Charset.forName("UTF-8"));
...
```

En otros casos, nos puede interesar escribir un registro desde un array de cadenas (String[]). Para ello necesitaremos usar el método ".writeRecord()";

```
...
  // ********************************
  // Para escribir registros completos
  // ********************************
  String[] datos = new String[2];

  datos[0]="1";
  datos[1]="Juan";
  csvWriter.writeRecord(datos);

  datos[0]="2";
  datos[1]="Pedro";
  csvWriter.writeRecord(datos);

  datos[0]="3";
  datos[1]="Inés";
  csvWriter.writeRecord(datos);


...
```

13.3.- Leer CSV desde Java

Vamos a leer el siguiente archivo CSV denominado "**productos.csv**":

```
idpro|descrip|precio
1|productoA|50.0
2|productoB|90.0
3|productoC|40.0
```

Tendremos una parte para crear el objecto de la clase **CsvReader** y luego otra para leer la cabecera y los registros.

```
public static void main(String[] args) {
  try {
    CsvReader products = new CsvReader("./datos/productos.csv");
    products.setDelimiter('|');
    products.setRecordDelimiter('\n');

    products.readHeaders();

    while (products.readRecord()) {
      String proID =      products.get("idpro");      //products.get(1)
      String proDescrip = products.get("descrip");    //products.get(2)
      String proPrecio =  products.get("precio");     //products.get(3)

      System.out.println(proID + "|" + proDescrip + "|" + proPrecio);
    }
    products.close();

  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```